Toward Practical Public Key Anti-Counterfeiting for Low-Cost EPC Tags

Alex Arbit, Yossef Oren and Avishai Wool
Computer and Network Security Lab, School of Electrical Engineering
Tel-Aviv University, Ramat Aviv 69978, Israel
Email: {alexand5|yos|yash}@eng.tau.ac.il

***Abstract*—In this work we report on a practical design, and a working prototype implementation, of a public-key anti-counterfeiting system based on the Electronic Product Code (EPC) standard for supply chain RFID tags. The use of public-key cryptography simplifies deployment, reduces trust issues between the tag integrator and tag manufacturer, eliminates the need for on-line checks by a central authority, and protects user privacy. Contrary to earlier claims of impracticality, we demonstrate that EPC tags are capable of performing full-strength public-key encryption. The crucial element in our system is WIPR, a recently-proposed variant of the well known Rabin encryption scheme, that enjoys a remarkably low resource footprint (less than 4700 gate equivalents for a complete ASIC implementation) – for a full-strength 1024-bit encryption. Our prototype system consists of an ultra-high frequency (UHF) tag running custom firmware, which communicates with a standard off-the-shelf reader. No modifications were made to the reader or the air interface, proving that high-security anti-counterfeiting tags and standard EPC tags can coexist and share the same infrastructure. Surprisingly, we identify that the time bottleneck is not the tag's computation time: the delay is dominated by inefficiencies in the way the reader implements the EPC standard. The insights from our performance measurements let us identify how a few simple changes to the reader can drastically improve the system throughput.**

Keywords: Anti-counterfeiting, security, EPC

I. Motivation

Counterfeiting is one of the major problems of modern commerce. According to a 2008 OECD study[11], the market value of counterfeit goods sold every year amounts to over 600 billion dollars, or 5%-7% of total world trade. The introduction of electronic RFID tagging to various goods, from medicines to banknotes, offers a chance to defeat counterfeiting by attaching a verifiable unique ID to high-value goods. As noted in [22], the EPC network is a natural match to anti-counterfeiting efforts since it provides each tagged item with a unique ID and has infrastructure in place to track and trace any tag as it travels through the supply chain.

Each individual tag in the current generation of Electronic Product Code (EPC) tags has an unencrypted unique tag identifier (TID) which can be sent to readers as after performing an ordinary inventory request. As noted in [14], there are two main approaches to using this TID as an anti-counterfeiting measure. The first approach is the "track and trace", or e-pedigree approach. In this approach the tag's location is phyically tracked and updated in some central server every time it is accessed by a reader. A counterfeited tag will exhibit an unusual and detectable usage pattern, such as being present in two different countries at the same time. Sophisticated software running on a central server and monitoring all tags worldwide can then detect these anomalies and prevent counterfeiting. The main disadvantage of this approach is its significant communications and infrastructure overhead which stems from the fact that the readers must constantly communicate with an online central server. As noted in [22], even if only a single uncooperative party in the supply chain does not properly update the central server the entire system quickly loses its effectiveness. Another problem of the trace-and-trace approach is the complete loss of privacy for individuals carrying the tagged items, who will now be tracked centrally together with the items they carry. This loss of privacy is problematic (and sometimes illegal) when dealing with items such as banknotes or medicine.

The second approach to RFID-based anti-counterfeiting is based on cryptography. In this approach each tag has, in addition to its TID, a certain additional "secret value", or payload, which is disclosed to the reader as proof of its authenticity. This payload is not sent in the clear over the air, since adversaries may clone the tag by observing over-the-air protocol exchanges. Instead, some form of encrypted communications is used to present the payload to the reader, which then verifies it using cryptography. By applying a public-key signature to this payload we can prevent an adversary who knows one tag's payload (obtained, for instance, by reverse engineering one tag) from creating a different valid payload. The fact that the tag can prove its own identity without relying on a central server to verify it allows a more flexible use of offline and semi-offline readers. Additionally, the privacy of tag-bearing individuals can be preserved by limiting the amount of information disclosed by the unencrypted element of the protocol exchange and keeping all sensitive information as part of the encrypted payload.

Supported in part by grants from Check Point and from Giesecke & Devrient.

Methods of anti-counterfeiting based on cryptography are further divided by the choice of either symmetric (also known as secret key) or asymmetric (also known as public key) cryptography to secure the communications between tag and reader. In symmetric cryptography the same cryptographic key exists both on the reader an on the tag. This key allows both encryption and decryption of the secret payload. In contrast, asymmetric ciphers allow a less sensitive public key (which can only encrypt but cannot decrypt other messages) to be stored on the tag, while the high-security private key (which can both encrypt and decrypt) is only stored at the reader side. Recent results in hardware design of symmetric ciphers indicate that cheap 5-cent tags may be powerful enough to support strong symmetric cryptography, but any secret-key-based system's security breaks down completely if the key is recovered from even a single tag. This is an acute problem in the supply chain environment, where tags are created in very large quantities by myriad untrusted parties and their use is relatively uncontrolled. In fact, several test implementations of secret-key-based anti-counterfeiting use a different secret key for each individual tag. Since it is very difficult to distribute all keys in advance to all readers, the key database must be stored on a secure central server which must be online at all times, incurring an even higher communications overhead when compared to the track-and-trace approach[14]. The case for public-key cryptography in EPC tags is thus very strong. However, public key cryptography was considered out of reach for general purpose tags due to its high hardware cost [1], [9].

A problem common to any cryptographic-based approach is the fact that it will have to be compatible to some extent with the EPC C1G2 specification[5]. In contrast to the track-and-trace approach, which appears over the air to be essentially identical to any other tag inventory command, a cryptographic anti-counterfeiting protocol will involve the additional protocol step of sending the payload and verifying it. Since the C1G2 specification does not specifically include an anti-counterfeiting protocol, it was unclear how the existing infrastructure will be able to support such a system[3].

In our work we set out to deliver a working implementation of public key-based anti-counterfeiting for EPC tags. First, we selected a public-key scheme which could offer suitable security properties while still staying within the stringent hardware requirements of low cost EPC tags. Next, we investigated the systems engineering challenges – the new scheme must be applied within an open system, consisting of mutually distrustful parties with conflicting interests, and it must integrate with the existing standards and infrastructure of tags, readers and backend systems. Finally, we implemented the scheme and measured its performance on real EPC equipment. We believe our system addresses all challenges and is usable with the current generation of hardware and standards.

The fundamental elements of the RFID environment in general are the **tag** and the **reader**, connected by a broadcast wireless medium. As mentioned previously, the tag bears a secret ID, or **payload**, which it wishes to provide to the reader in a secure manner. Looking beyond the tag-reader association, there are several additional parties in any RFID supply-chain environment whose goals must also be considered. The **tag integrator**, or supply chain owner, is the party whose security and privacy goals must be realized, but some of these goals clash with the immediate objectives of the reader itself (for example, a rogue reader would like to be able to create counterfeit tags, but the tag integrator does not want this to happen). Another significant party is the **tag manufacturer**, whose role is to provide usable tags, preferably with minimum communication or commitment to the tag integrator. These two parties have limited trust in each other. In particular, cryptographic keying creates a hurdle – providing the relevant cryptographic keys to the relevant parties, tracking the use of these keys and allowing revocation when the keys fall into undesirable use.

Established results show that a tag essentially has no secrets once it falls into the hands of a moderately-funded adversary[18]. While a physically captured tag can be perfectly cloned, we wish to limit the effect of such a compromise on the security of the system as a whole. In particular, if the tag carries no secret keys, and its ID is cryptographically signed, then the adversary can only clone the captured tag – but cannot counterfeit other tags.

A. Our Contribution

In this work we present a working implementation of a supply-chain system which includes a public key-based anti-counterfeiting function while staying compliant with the EPC C1G2 standard and supporting off-the-shelf RFID readers. Our solution addresses both the cryptographic aspects and the systems engineering aspects of the anti-counterfeiting system. The cryptographic scheme we selected is WIPR, an ultra-low-power public key cryptosystem developed by Oren and Feldhofer[19], which realizes strong 1024-bit Rabin encryption with a minimal hardware footprint of 4700 gate equivalents for a complete ASIC implementation. Other than a security module that implements the anti-counterfeiting function, the reader side is identical to a standard non-authenticated RFID system. Specifically, the reader hardware we used was not modified in any way and the over-the-air communication used only mandatory commands from the EPC C1G2 command set[5]. The use of public-key cryptography, as opposed to secret-key cryptography, makes the system to much simpler to deploy since it can operate without a constant online connection to a central server.

The rest of the document will be arranged as follows: first, we survey related work in the field. Section II discusses the operational advantages of using public-key cryptography for anti-counterfeiting. Sections III and IV describe our system and the results of our implementation.

Section V concludes with some open issues and directions for future research.

B. Related Work

The suitability of the EPC network for anti-counterfeiting was first explored in [22] and [12]. These works suggested adding a central anti-counterfeiting server to the EPC network and did not explicitly cover the use of encryption.

Much additional work was performed under the European BRIDGE project[1]. The BRIDGE project evaluated several approaches to providing RFID-based anti-counterfeiting, both based on cryptography and on a centralised tracking approach[14]. One of the striking conclusions of the work performed by this project was that cryptography, and specifically secret-key cryptography, was less cost effective for tags than using plaintext tags which are tracked and traced as they move between readers. The main reason cited in [14] for this result was the heavy server-side burden of storing and managing many secret keys, causing a computational load estimated to be ten times as much as the already formidable load caused by conventional track-and-trace. In this work we challenge some of their conclusions, by showing that public-key cryptography is quite feasible and that it requires minimal central computing power.

Low resource implementations of secret-key cryptosystems, the most noteworthy of which is AES [6], have already been demonstrated on physical chips. Low-resource public key cryptosystems have yet to achieve this level of market readiness. The Rabin cryptosystem was first implemented in a low-resource setting by [9]. The low cryptographic security and high hardware cost offered by the authors' unmodified Rabin implementation (512-bit encryption in 16,700 gates) led them to declare that this cryptosystem is unsuitable for RFID tags, a belief that was echoed by many other subsequent works. Other public-key RFID contenders can be found in works such as [7], [8]. These implementations generally require more gates than can fit in a low-cost 0.35 µm process tag or rely on uncommon features such as very large random sources. Of special note is the Crypto-GPS scheme presented in [16]. While this scheme has a potentially low hardware cost, it is by design a zero-knowledge *identity-proving* scheme and cannot be used securely in an *encryption* setting. Another survey of public-key schemes based on elliptic-curve cryptography can be found in [13].

The public key scheme we use in this work is WIPR, which is described in [19]. WIPR is based on a randomized variant of the well-known Rabin cryptosystem[20], first discussed in [10]. It is well suited for our solution because it has one of the smallest hardware footprints and largest payload capacities of all published high-security public-key schemes. This scheme's applicability to low-resource smartcards was first explored in [17], [21]. The properties of this cipher are presented in Table I.

Cipher Strength	1024 bits
Challenge size	80 bits
Response size	2208 bits
Payload capacity	864 bits
Area	4682 GEs
Total power consumption	14.2µA

Table I
PROPERTIES OF THE ASIC IMPLEMENTATION OF THE WIPR CIPHER[19]

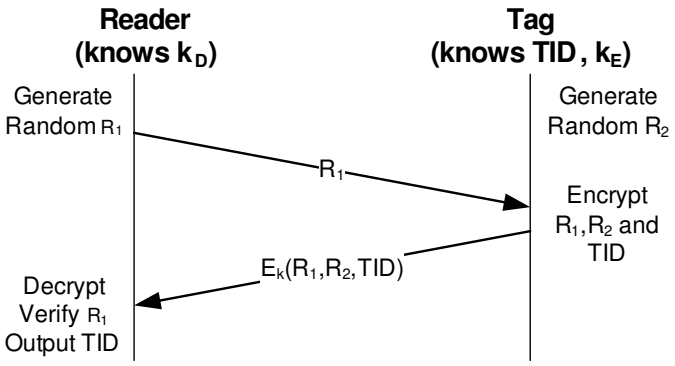

Figure 1. A general cryptographic anti-counterfeiting protocol

There are several additional tag-to-reader authentication protocols presented in other works, based on bitwise operations, pseudo-random numbers, hash-functions and physically unclonable functions (PUFs). For a survey covering these protocols, please refer to [15].

II. The Advantages of Public-Key for RFID Anti-Counterfeiting

The general structure of a cryptographic anti-counterfeiting scheme based on a challenge-response protocol is illustrated in Figure 1. As shown in the figure, each tag contains a secret payload (most commonly a cryptographically signed version of the TID with some additional sensitive information) and a cryptographic key it uses to communicate with the reader. To begin the protocol the reader sends the tag a random challenge, which need not be encrypted. The tag then constructs a message consisting of this challenge, internally generated random bits and its TID and encrypts this message under the cryptographic key. Finally, the tag sends the encrypted message to the reader. The reader then verifies that the response contains the challenge it had originally sent and that the secret payload is valid.

It is important to note that this protocol is used both for symmetric and assymetric cryptographic schemes. The only difference lies in the choice of keys delivered to the participating parties – in a symmetric scheme both sides receive the same key and can both encrypt and decrypt messages. In contrast, in an assymetric scheme tags can only encrypt messages, but readers can both decrypt messages and validate signatures. This is an important distinction since tags are relatively easy to reverse-

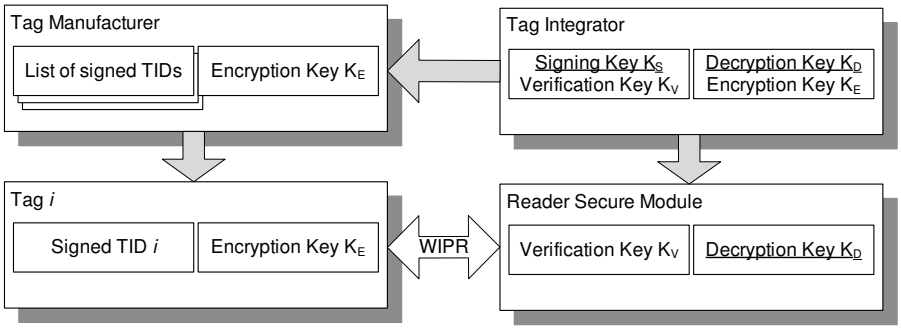

Figure 2. A logistic view of the suggested public-key based anti-counterfeiting system. Private keys (signing and decryption) are underlined.

engineer[18]. This means that an adversary needs only to attack a single symmetric-key tag to compromise the system. To protect against this fact, symmetric-key anti-counterfeiting schemes typically distribute many secret keys, some going as far as specifying a unique secret key for every tag. This added data requirement causes a heavy logistical burden on the system, which was estimated in [14] to be ten times as heavy as a complete track-and-trace protocol. In contrast, a public key-based system remains secure even if the adversary has complete knowledge of the entire system other than the private key. Since tags do not require decryption capabilities, they never store a private key: the private keys are only kept on the more secure readers. This fact lets the reader operate without a constant online link to a central server.

The WIPR scheme is a straightforward implementation of the challenge-response protocol presented earlier. In contrast to other low-resource authentication schemes, during the execution of the protocol the data on the tag is not rewritten, and the tag can be used for an unlimited amount of times. For a detailed description of the WIPR scheme, please refer to [19].

In some cases it is desirable to prevent the exact identity of tags from being determined by unauthorized readers. In other cases, merely discovering that a certain tag belongs to a certain batch (for example, medicine or high-denomination banknotes) poses a security risk. Our public-key scheme can enable both of these properties if the tag's public EPC value is set to a fixed or semi-random value which cannot identify the tag indepdently, while the real EPC is made part of the secret payload.

A logistic view of the suggested public-key based anti-counterfeiting system is described in Figure 2. The figure represents the various members of a simple tag-equipped supply chain – the **tag manufacturer**, which produces and deploys many tags for a multitude of clients, the **tag integrator**, which wishes to use anti-counterfeiting technology but does not manufacture them itself, and finally the **reader**, which performs the actual authentication process. The tag integrator creates two pairs of public-private keys: a **private signing key** together with its **public verification key**, and a **private decryption key** together with its **public encryption key**. The signing key never leaves the premises of the integrator. Instead, the integrator generates a long list of signed TIDs and sends them to the manufacturer together with the public encryption key. Without the private signing key the tag manufacturer is unable to create arbitrary signed TIDs, a fact which reduces the amount of trust and liability required in the system. The tag manufacturer then generates a multitude of tags, each with the public encryption key and with an individual signed TID from the list. The use of public key cryptography guarantees that an adversary gains no private information from reverse engineering an individual tag other than the tag's TID.

The reader receives the private decryption key and the public verification key from the tag integrator, but not the private signing key. Thus, it is able to securely verify the identity of tags but is unable to forge new ones. Significantly, once these two keys are delivered to the reader (probably in the form of a secure module such as a smart card), the system can operate completely offline and does not require constant communication with a server.

III. Lab Setup

The system we built consists of an EPC C1G2-compliant RFID tag, an EPC C1G2-compliant RFID reader, and two PC workstations.

The system setup is presented in Figure 3. Our system used the UHF Demotag, a hardware prototyping platform developed by IAIK TU Graz. As stated in [2], the tag is battery-powered, but behaves like a fully-passive tag in the reader field. It is fully compatible to ISO 18000-6c and EPC Class1 Generation2 standards. The tag is optimized for easy adaptability to allow easy and fast development of prototypes. It features a ATMega128 microcontroller with JTAG and ISP interface for programming. An RS232 interface is available for configuration and logging. The front-end consists of discrete devices on a PCB, with a PCB antenna that is tuned to 868MHz. The tag is connected via a serial RS232 communication link to a Linux workstation running the CrossStudio for AVR embedded development environment by Rowley Associates, version 1.4. The firmware executes on power-on from Atmega128 on-chip Flash memory. The CAEN

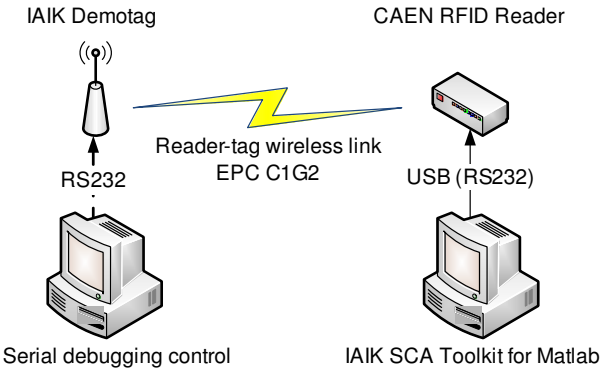

Figure 3. System setup

RFID DK828EU reader is used as the tag's interrogator. It features an controller module with embedded EPC C1G2 reader firmware which is controlled via USB link by a Windows workstation running Matlab. The DK828EU reader conforms with European ETSI power requirements[4]. In our lab tests we found this reader has an average read rate of approximately 15kbps, a fact which dominated the overall performance of our system. The IAIK SCA Toolkit provides the connection between the reader's software libraries and Matlab. Finally, an RFID wireless link is established between the demotag and the reader.

Figure 4 demonstrates the full WIPR Protocol flow through EPC C1G2 air interface using standard EPC protocol commands. The reader first sends the standard INVENTORY command. WIPR tags do not respond to this command with the full EPC, which may be sensitive and should not be disclosed. Instead, the tag sends a special EPC value indicating that it is a WIPR tag and possibly disclosing a limited subset of the EPC which is sufficient for use with non-secure readers. To allow for a single WIPR tag to be succesfully singulated when multiple WIPR tags are present, part of this special EPC value will be a random value computed on boot. The reader then starts sending the 80-bit cryptographic challenge. This operation is performed through the standard EPC C1G2 WRITE command. After the challenge is sent, the tag automatically encrypts its payload of data (consisting of the TID, the challenge, and locally generated random bits) and places it in the SRAM buffer on ATMega128 chip ready to be sent by demand. Once the reader issues a standard BLOCK_READ command to the tag, the ciphertext is read out from the tag. The reader is free to initiate as many cycles of data transfer as it wishes between 1 and 138 16-bit words (the entire encrypted payload). As shown in the following subsection, larger block sizes result in a faster and more efficient data transfer.

It is important to note the three times marked in Figure

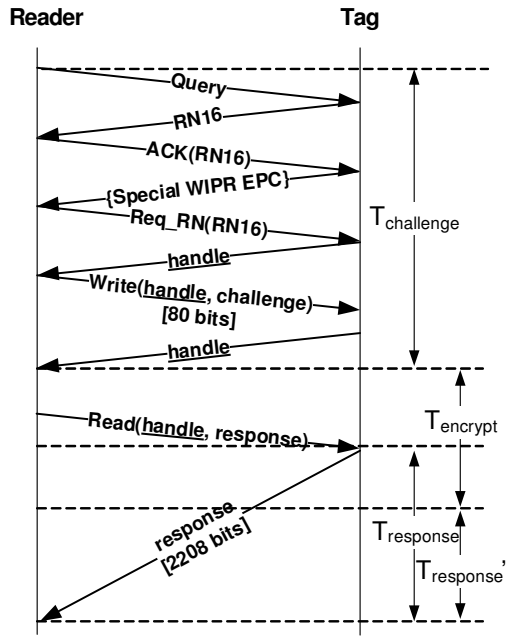

Figure 4. The full WIPR implemented using mandatory C1G2 commands (based on [5, annex E])

4 as $T_{challenge}$, $T_{encrypt}$ and $T_{response}$. While $T_{challenge}$ and $T_{response}$ are determined by the speed of the link between the tag and the reader, $T_{encrypt}$ is solely a function of the implementation quality of the WIPR algorithm. It can also be noted that only a part of $T_{response}$ (marked as $T'_{response}$) happens after encryption is completed. As we discuss in the following subsection, this is due to a special property of the WIPR algorithm which allows for the ciphertext to be generated byte by byte.

IV. Implementation Results

Implementing cryptographic hardware for passive RFID tags is challenging due to the fierce constraints. The main objectives for the designed hardware are to minimize power consumption and to reduce the necessary chip area. The reason for the low-power constraint is the operating range, since the power available to a UHF RFID tag goes down as the square of the distance between the tag and the reader. The second big issue is the chip area. The cost of an RFID tag increases linearly with the die size. In our implementation both Flash and SRAM memory are available up to a reasonable amount for RFID tag usage size because of the chosen implementation method using embedded microcontroller. Thus, we came to achieve two main goals: the best performance possible resulting in a best cryptographic algorithm execution time on a given microcontroller, and lowest possible resources usage to make the WIPR public key cryptographic scheme largely affordable in terms of hardware gate count and costs for any chosen implementation method.

Implementing the WIPR scheme had a very minor effect on the resource cost of the IAIK demotag. The code section

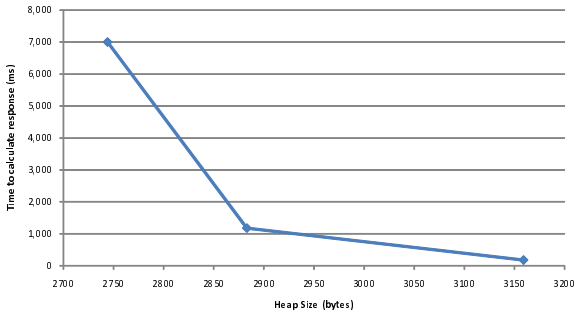

Figure 5. $T_{encrypt}$ as a function of heap size

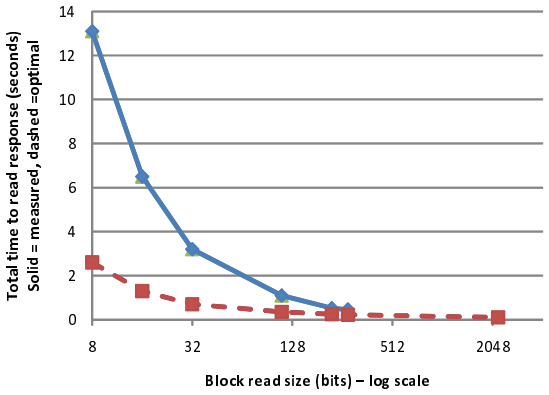

Figure 6. $T_{response}$ as a function of block read size

of a firmware design with the complete WIPR implementation requires 33540 bytes, only 7.5% (2534 bytes) more than the standard version of the firmware without WIPR support. WIPR uses only 660 bytes of the available 4KB of SRAM in its most RAM-heavy implementation.

To obtain a better understanding of system's resources usage we now give a brief description of the WIPR scheme, based on [19].

The tag is provided with a 1024-bit public key n, which is stored in the tag's ROM and can be copied to the heap on boot to improve performance. The tag also stores its signed TID, which can be up to 864 bits long (for reference, a high-security ECDSA signature is 320 bits long). When issued with a fresh challenge, the tag generates two random bit strings $r_{t,1}$ (80 bits) and $r_{t,2}$ (1104 bits).

When the tag receives the challenge r_r sent by the reader, it stores it in heap memory. It then creates its response message $P = r_r||r_{t,1}||TID$ – i.e., $r_{t,1}$ is used as random padding to bring the plaintext to 1024 bits. Beginning at the least significant byte, the encrypted message $M = P^2 + r_{t,2} * n$ is computed using multiplication by convolution. Note that there is no modular reduction, so the message M is 2208 bits long. The response bytes are then stored in SRAM memory. The WIPR algorithm structure allows encryption in a byte-by-byte on demand fashion, supporting devices with limited memory and also allowing the response to be generated in the background.

Three possible scenarious were evaluated: first we evaluated a naïve implementation which does not cache the values of P and $r_{t,2}$ values in SRAM prior to convolution but instead recalculates them on demand. Next, we tried caching the value of P before convolution. Finally, we tried caching the values of both P and $r_{t,2}$. As depicted in Figure 5, caching data on the heap has a dramatic effect on the execution time. The first scenario required 7 seconds to encrypt. The second scenario (caching only P) took 1.18 seconds, while the third scenario (caching both values prior to convolution) sped the calculation to 180 milliseconds. The convolution was implemented using the ATMega128's built-in hardware multiplier for all scenarios.

The results show that adding a small amount of heap space can improve tag performance by several orders of magnitude. The decision to use SRAM resources is easy when a microcontroller is chosen for the task. However, one should keep in mind that SRAM is expensive in space for ASIC implementations, so a cost-performance trade-off should be considered for each implementation method.

Figure 6 shows the value of $T_{response}$ as a function of the amount of bits accessed in each block read operation. Recall that the computed result of 2208 bits is read from the tag in a sequence of BLOCK_READ operations, and the block size is an implementation parameter. If a single 16-bit word is read in every round trip, the 138 read commands issued by reader take 6.5 seconds to transfer the entire payload. On the other hand, a block size of 34 bytes (272 bits, the maximum size supported by our lab setup) allows the same payload to be transferred in only 0.46 seconds using 8 block reads. Upon further investigation, we found that the system's bottleneck is concentrated in the CAEN reader firmware, which takes about 40ms to perform a single read operation, regardless of the size of the data exchanged. This happens because the reader performs a fresh singulation protocol each time a tag is accessed, even if the tag is already in the SECURED state. The singulation process results in 3 unnecessary protocol round-trips per command, dramatically reducing the I/O performance. The reader we surveyed also powers up the radio circuit before each command and shuts it down again after the command concludes, further reducing performance. The dashed line in Figure 6 shows the estimated performance of the same reader assuming the tag enters the read process powered on and singulated and does not repeat the singulation protocol between commands.

The current reader's configuration did not allow us to interfere in its order of execution or implement any protocol optimization. Table II estimates the values of $T_{response}$ for a reader-tag link using an optimized EPC C1G2 flow. The estimation assumes the fixed cost of 40ms related to powering up and singulating the tag was already incurred when the challenge was sent, so all time incurred

Ciphertext bytes read per block	Measured $T_{response}$ (sec)	Estimated $T_{response}$ (sec)
1	13.1	1.02
2	6.5	0.57
4	3.2	0.34
14	1.1	0.18
28	0.52	0.15
34	0.46	0.14
276	unsupported	0.12

Table II
$T_{response}$ AS A FUNCTION OF BLOCK READ SIZE

Protocol Step	Current results	Partial pipelining	Full pipelining	Optimization step
$T_{challenge}$	200	85	85	Write all 80 bits of the challenge in a single round-trip
$T_{encrypt}$	180	180	180	
$T_{response}$	460	180	112	Keep tag alive and singulated
$T'_{response}$	460	60	0	Pipeline encryption and transmission (via FIFO or via background calculation)
Total	840	325	265	

Table III
PERFORMANCE OF THE COMPLETE WIPR PROTOCOL UNDER VARIOUS OPTIMIZATIONS (ALL TIMES ARE IN MILLISECONDS)

is related to the propagation delay of BLOCK_READ operations performed at 15kbps.

A. Optimizations

The results we measured are for a completly serialised operation, with the transmission of the ciphertext starting only after the last byte of ciphertext is calculated ($T_{response} = T'_{response}$). In addition, the current firmware of the demotag supports writes of no more than 2 bytes and reads of no more than 34 bytes, resulting in 5 commands for writing the challenge and at least 8 for reading the response. Finally, the off-the-shelf reader we evaluated communicates with tags in an inefficient way, as discussed previously. By implementing relatively minor tweaks to these limitations, we can dramatically improve the operation of the system. Table III shows the estimated performance gains of these optimization steps.

The first and immediate improvement could be achieved by better use of the air interface. By sending the challenge in a single 80-bit packet and keeping the tag in the SECURED state, we can reduce $T_{challenge}$ from 200ms to an estimated 85ms. Next, we can remove the unnecessary singulation steps by making sure the reader keeps the tag powered on and in the SECURED state throughout the response phase. In addition, we can pipeline the encryption and response transmission: using WIPR, the tag can compute the ciphertext in 34-byte blocks and send them to the reader as soon as they are ready. The total time to perform the entire protocol in this case is equivalent to the time required to power on the tag and send it a challenge (85ms), the time required for the tag to calculate the full response (180ms) and the time required to send the final 34-byte chunk, which is ready only after encryption is finished (60ms). Under these minor modifications we estimate the entire protocol (including both identification and authentication) will take 325ms.

For a more dramatic optimization, we can read the entire 276-byte response in a single read command which is issued immediately after the challenge is sent. This is possible since the tag can be designed to concurrently backscatter the initial bytes of the ciphertext while it calculates the following ones. Since the data link takes only 112ms to transfer 2208 bits, the entire protocol time is dominated in this case by $T_{encrypt}$, leading to a total estimated time of 265ms for the entire protocol.

V. Discussion and Future Work

Public-key cryptography was previously claimed to be impractical for RFID tags. The reasons for this claim were the high cost (in gate count and power consumption) of public key encryption, its slow performance when compared to secret-key ciphers or hash functions, and finally the lack of standards support. WIPR was shown in [19] to have an acceptable gate count and power consumption, but the time presented in [19] was 600ms per encryption, a delay which might be considered too much in a supply-chain scenario. In our work, which presents a working system, we wanted to discover whether the cryptographic operation is indeeed an inherent bottleneck or whether it can be sped enough to make the system usable. We also wanted to address the system issues and find out whether a practical public-key system can be created using today's hardware and standards.

We considered the general-purpose 8-bit microcontroller present on the demo tag to be inherently slower than a custom designed ASIC implementation. Indeed, a naïve software implementation of the WIPR protocol which was functionally identical to the ASIC's implementation took an unacceptable 7 seconds to perform an encryption. However, as illustrated in Figure 5 the addition of RAM significantly sped up the software implementation to the point that the entire encryption took 180ms. It will be interesting to generalize those results beyond the hardware used for the implementation and find out whether the ASIC implementation of [19] can be similarly sped up by adding a small amount of RAM. It will also be interesting to consider integration issues between the current EPC C1G2 tag chips (e.g., the Alien Higgs-3 or the Impinj Monza 4) and the suggested anti-counterfeiting function.

We found that another serious bottleneck is in communication, with the dominant parameter being the amount of roundtrips made by the reader. This problem is even more acute if the reader being used does not recognize

the concept of sessions and repeats the singulation process with the tag every time it wishes to send it a command. It will be interesting to investigate whether other reader vendors handle multi-request sessions to a single tag more efficiently. If the tag can calculate the response bits faster than they are transmitted, optimal performance can be achieved by a pipeline design which transmits the ciphertext byte by byte as it is being generated within the context of a single large read command. This results in a very efficient performance and a saving of valuable RAM. Even when using minimal optimizations, the time required for the complete protocol is quite reasonable (≈325ms), especially considering the fact that there are no additional communications with a central server.

The scheme presented here is designed for fully offline operation. While the use of public key cryptography prevents wholesale counterfeiting, an adversary who is able to clone a tag will still be able to create many counterfeit products equipped with copies (clones) of the same captured "original" tag. Local readers, which do not perform on-line verifications (off-line verification is one of the main reasons to use public-key schemes), will recognize the counterfeit products equipped with the cloned tag as genuine. To protect against this threat our scheme can be further strengthened either by the use of standard track-and-trace or by periodically distributing a small "blacklist" of cloned TIDs (we expect only a small amount of these cloned TIDs, since they can only be extracted by reverse engineering a captured tag).

Our work shows a working implementation of a public-key based anti-counterfeiting system based on the EPC C1G2 standard. The system consists of a tag running custom firmware, but communicates with a standard off-the-shelf reader using mandatory EPC C1G2 commands. The fact that our system uses public-key cryptography simplifies deployment and trust issues and should drastically reduce the system costs related to the use of anti-counterfeiting. No modifications were made to the reader or the air interface, proving that high-security anti-counterfeiting tags and standard EPC tags can coexist and share the same infrastructure.

We conclude that the public-key approach is a viable design alternative for anti-counterfeiting RFID EPC tags.

Acknowledgements

The authors wish to thank the anonymous reviewers for their detailed and helpful remarks.

References

[1] M. Aigner, T. Burbridge, J. J. Cantero, A. Ilic, J. Al-Kassab, O. Kasten, and A. Plaza. Security technology roadmap. Technical report, BRIDGE Project, June 2009.

[2] M. Aigner, T. Plos, and A. R. S. Coluccini. Secure semi-passive RFID tags – prototype and analysis. Technical report, BRIDGE Project, November 2008.

[3] D. V. Bailey and A. Juels. Shoehorning security into the EPC tag standard. In R. D. Prisco and M. Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006, LNCS*, pages 303–320. Springer, September 2006.

[4] H. Barthel. UHF RFID regulations. Online, June 2006.

[5] Epcglobal inc., EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz – 960 MHz, version 1.0.9. Online, September 2005.

[6] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In J.-J. Q. Marc Joye, editor, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop, LNCS*, volume 3156, pages 357–370. Springer, July 2004.

[7] M. Finiasz and S. Vaudenay. When stream cipher analysis meets public-key cryptography. In E. Biham and A. M.Youssef, editors, *Selected Areas in Cryptography - 13th International Workshop, SAC 2006, LNCS*, volume 4356, pages 266–284. Springer, September 2007.

[8] J. Furbass, F.; Wolkerstorfer. ECC Processor with Low Die Size for RFID Applications. *IEEE International Symposium on Circuits and Systems, 2007*, pages 1835–1838, 27-30 May 2007.

[9] G. Gaubatz, J.-P. Kaps, E. Ozturk, and B. Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 146–150, March 2005.

[10] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of Computing*, pages 365–377, New York, NY, USA, 1982. ACM.

[11] W. Hübner. *The Economic Impact of Counterfeiting and Piracy.* OECD Publishing, June 2008.

[12] R. Koh and E. W. Schuster. Securing the pharmaceutical supply chain. Technical report, MIT Auto-ID Center, June 2003.

[13] Y. K. Lee, L. Batina, D. Singelee, B. Preneel, and I. Verbauwhede. Anti-counterfeiting, untraceability and other security challenges for RFID systems: Public-key-based protocols and hardware. In D. Basin, U. Maurer, A.-R. Sadeghi, and D. Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 237–257. Springer Berlin Heidelberg, 2010.

[14] M. Lehtonen, J. Al-Kassab, F. Michahelles, and O. Kasten. Anti-counterfeiting business case report. Technical report, BRIDGE Project, December 2007.

[15] M. Lehtonen, T. Staake, and F. Michahelles. From identification to authentication - a review of RFID product authentication techniques. In D. C. Ranasinghe and P. H. Cole, editors, *Networked RFID Systems and Lightweight Cryptography*, pages 169–187. Springer Berlin Heidelberg, 2008.

[16] M. McLoone and M. Robshaw. Public key cryptography and RFID tags. In M. Abe, editor, *Topics in Cryptology – The Cryptographers' Track at the RSA Conference 2007, LNCS*, volume 4337, pages 372–384. Springer, February 2007.

[17] D. Naccache. Method, sender apparatus and receiver apparatus for modulo operation. European patent application no. 91402958.2, Filed 10/27/1992.

[18] K. Nohl and H. Plötz. MIFARE – little security, despite obscurity. Technical report, 24th Chaos Communication Congress, December 2007.

[19] Y. Oren and M. Feldhofer. A low-resource public-key identification scheme for RFID tags and sensor nodes. In D. A. Basin, S. Capkun, and W. Lee, editors, *WISEC*, pages 59–68. ACM, 2009.

[20] M. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, MIT, Cambridge, MA, USA, 1979.

[21] A. Shamir. Memory efficient variants of public-key schemes for smart card applications. In A. D. Santis, editor, *Advances in Cryptology – EUROCRYPT '94, LNCS*, volume 950, page 445. Springer, January 1995.

[22] T. Staake, F. Thiesse, and E. Fleisch. Extending the EPC network: the potential of RFID in anti-counterfeiting. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 1607–1612, New York, NY, USA, 2005. ACM.